SIM Testbed 3 real-time control software

Elizabeth A. McKenney °, Oscar S. Alvarez-Salazar#

Jet Propulsion Laboratory, 4800 Oak Grove Dr, Pasadena, CA 91109

ABSTRACT

SIM System Testbed 3 (STB3) features three optical interferometers sharing a common baseline, as a dynamic representation of the SIM instrument. An artificial star feeding the interferometers is installed on a separate optics bench. All three interferometers use photons captured by avalanche photo diodes (APDs) to measure the position and quality of fringes, and additional pointing precision is achieved by fast steering mirrors (FSMs) that keep the star images centered on the beam combining optics using a CCD camera. Each interferometer uses internal metrology to measure changes in its optical pathlength. External metrology beams measure changes in the baseline vector.

This system acquires and tracks white light fringes with one interferometer, while the other two acquire and track laser light fringes representing the bright guide stars that will be used by SIM. The white light source represents a dim star that cannot supply enough photons for the Science interferometer to lock onto fringes in closed-loop mode; instead it operates open-loop, using pathlength corrections fed to it from the two guide interferometers and the external metrology subsystem to reject disturbances and maintain the fringes. This tracking mode is known as Pathlength FeedForward (PFF).

The precise real-time behavior required to achieve this result is implemented by a complex set of interacting software control loops. This paper describes how these loops take advantage of the benefits of the RTC Core architecture, and how they work together to accomplish STB3's objectives.

Keywords: Interferometry, atmospherics, real-time control, object-oriented, testbed

1. INTRODUCTION

JPL's Space Interferometry Mission (SIM), scheduled for launch in 2009, will determine the positions and distances of stars from an earth-trailing solar orbit, several hundred times more accurately than previous programs. This level of accuracy will allow SIM to determine the distances to stars throughout the Galaxy and to probe nearby stars for Earth-sized planets[1].

Part of SIM's science goal involves the study of "dim" stars (generally less than 12th magnitude), which require very long integration times in order to collect sufficient data. These integration times can be in the tens or even hundreds of seconds. The observing instrument must maintain a precise lock on the star during that time, compensating invisibly for vibrations induced by the spacecraft or internal errors such as thermal drift. This is made more difficult by the fact that the instrument receives fewer photons from a dim star, which lowers the SNR of the signal and reduces the effective bandwidth of the system.

To combat these issues, the SIM instrument design contains three interferometers that share a common baseline. Two of them act as guides; during an observation they will point at bright stars where it is easier to maintain a precise lock. The third is the Science interferometer, which will point at the dim star of interest. As the two guides make adjustments to compensate for disturbances, they will feed forward their actions to the Science interferometer. The Science instrument will run virtually open-loop, taking the information it receives from the guides and compensating in a similar fashion as it records data from the dim star.

° Send correspondence to: M/S:171-113, Phone: 818 354 5136, Elizabeth.A.Mckenney@jpl.nasa.gov

Send correspondence to: M/S: 171-113, Phone: 818 393 5952, Oscar.Alvarez-Salazar@jpl.nasa.gov

The goal of SIM System Testbed 3 (STB3) is to demonstrate the feasibility of this "pathlength feedforward" or PFF technique under laboratory conditions, first on stiff optical tables and then again on a flexible 9-meter flight-like structure that more closely resembles the SIM instrument design. SIM's requirement for STB3 is to demonstrate fringe tracking with 10nm maximum optical pathlength jitter across the frequency spectrum.

2. SIM SYSTEM TESTBED 3 DESIGN

Interferometric fringes can only be achieved when the interfering light beams are perfectly matched with respect to both tilt and optical pathlength. The portion of the interferometer that adjusts the tilt is the *pointing* subsystem and the portion that accomplishes the pathlength matching is the *phasing* subsystem. The process of adjusting the pathlength in order to produce fringes where the beams intersect is called *fringe acquisition*. Finally, in order to stabilize the fringe once it has been found, the interferometer must continually adjust the tilt and pathlength to compensate for disturbances in the system; this is known as *fringe tracking*. Each of these tasks involves one or more real-time feedback loops, and in many cases the loops interact with one another through shared state.

The full STB3 system runs three interferometers simultaneously, but the software for each interferometer runs on a separate processor, communicating as needed via a shared memory card on the same backplane. All three processors load and execute the same object files to perform the basic functions of interferometry, but each is configured slightly differently to reflect differences in the hardware that it controls. The following is a description of the components of a single interferometer:

1. Two pointing software components (called Star Trackers, one for each arm of the interferometer) read a CCD image that contains the apparent star position for each arm of the interferometer. They then adjust fast-steering mirrors to move the star spots to specific locations in the image.
2. The phasing software component, called the Fringe Tracker, detects fringes by dithering the optical pathlength back and forth about a nominal position. Photons received by an avalanche photo diode are counted at several points in the dither waveform, representing variations in light intensity as the pathlength is varied. These photon counts are used to calculate fringe location, or phase, by means of a crude but effective heuristic. They are also used to calculate the SNR and visibility of the fringe signal.
3. For fringe acquisition the Fringe Tracker varies the nominal pathlength (i.e. neglecting any dither motion) according to a preset pattern, using the SNR and visibility values to indicate the possibility of fringes. During this time the fringe phase is calculated but is generally meaningless due to the low information content of the signal.
4. Once SNR & Visibility reach predefined threshold values, the acquisition stage is considered complete and the Fringe Tracker servo automatically changes behavior to track the fringe. It now uses the fringe phase to calculate how the pathlength should be varied to stay centered on the detected fringe.
5. The pathlength is varied in both fringe acquisition and fringe tracking by an optical component called a delay line. From a system-level view the delay line is simply a position actuator, however it accomplishes this goal using three different actuating mechanisms with vastly different properties and ranges of motion. The control system required to make the delay line behave in the desired manner is sufficiently complicated that it has its own separate software component (the Delay Line).

For 3-interferometer operation, the Science processor also loads an additional component called Pathlength Feedforward. This component collects the internal metrology and fringe phase information from the two guide interferometers, and uses it to calculate the open-loop position offset that the Science interferometer uses in PFF mode.

3. RTC CORE TOOLKIT

All of the software used in the STB3 experiment is based on the RTC Core Toolkit, a software package developed at JPL for use by several interferometer projects including SIM and STB3[2]. Over time, the toolkit has evolved into a more general-purpose architecture and set of core services applicable to a wide range of real-time embedded software control applications[3]. Its built-in capabilities include CPU management, configurable object definition, hardware driver management, inter-processor communication, precise periodic task scheduling, telemetry generation and filtering, and a digital servo and controller framework. Since it is object-oriented in design and implementation, a developer can easily implement the desired functionality for an application by deriving new classes that inherit and build on the generic functionality provided by the toolkit. RTC Core takes care of most of the lower-level details of developing a real-time control system, allowing the developer to spend his time working on the specifics of his control laws rather than periodic task scheduling or hardware interfaces.

The RTC Core toolkit's generic servo and controller framework includes a hard real-time periodic task scheduler, a servo container class that holds the specific control laws provided by the developer, and the ability to map the different control laws to a variety of system states, whether external (user-selected) or defined internally by the system. The servo/controller architecture in RTC has already been described in detail in a previous publication[2], but the following is a basic overview:

Any control software, whether open loop or closed loop, will involve some variation on the following: Reading values from sensors, applying some control law to the measurements, and commanding actuators that affect the system in a way that is observable by the sensors. The control behavior for a given high-level component – referred to in RTC Core parlance as a "Gizmo" – may need to vary during runtime, however, based on user commands or changing system conditions. (An example already mentioned from STB3 is how the Fringe Tracker component switches automatically from fringe acquisition behavior to fringe tracking behavior, based on the SNR and visibility of the fringe signal.) The RTC Toolkit allows the developer to write control laws for a variety of possible states of the system, and then attach them to a provided framework so that at runtime each of the control laws is mapped to the system condition that calls for it. As conditions change during system operation, the control laws are switched seamlessly behind the scenes as needed.

The RTC Core toolkit servo architecture supports two types of control law (controller) changes:

(a) User command, e.g. pressing a button on the GUI to change from Idle mode to Tracking ("servo mode").

(b) System conditions change, e.g. SNR of input signal decreases to a level where a different controller is needed ("controller mode"). An UpdateMode task runs periodically in the background, testing the system state and changing the controller mode automatically when appropriate.

In addition, a high-level component may have several controllers that operate simultaneously, possibly at different rates ("servo partitions"). Some or all of the partitions may need to change behavior when the servo mode or controller mode changes, and it may be essential for them to all change to the new mode at exactly the same time.

Together these form the three independent axes of a servo's control law array. In a given servo one or more of these axes may only have length 1, e.g. only one partition or one controller mode. However this design provides the flexibility for creating a high-performance control system capable of responding to a wide variety of system conditions. The following sections explain each of the software components of the STB3 system in more detail, describing how the real-time servo behavior of each one is mapped to the RTC Core Toolkit servo and controller framework.

4. POINTING SUBSYSTEM

The pointing subsystem keeps the two incoming light paths parallel, which is necessary for successful interferometry. Each interferometer in the STB3 experiment has two star tracker components that control the tilt of the beams. Each star tracker reads images from a CCD camera that is pointed along one arm of the interferometer, and calculates where the star spot falls on the image. It then calculates how to adjust a fast steering mirror in order to move the spot to the center of the image. Finally it sends the commands to the FSM actuators to make the adjustment. Outside the software, the FSM motion causes the spot to move on the CCD to close the loop.

The fine pointing control loop performs three steps:

1. Reads an image from the CCD camera and calculates the position of the star spot on the image.
2. Calculates the error between the star position and the desired target position, including PID compensation and an optional lag filter.
3. Sends the appropriate commands to the FSM to move in such a way that the star spot moves to the target position.

The Star Tracker control laws are organized into a single 50 Hz partition with 3 servo modes:

Off: Reads image and calculates centroid

Track: Reads image and calculates centroid, actuates FSM to drive star spot to center of image

Calibrate: Reads image and calculates centroid, dithers FSM for calibrating multi-actuator map

5. PHASING SUBSYSTEM: DELAY LINE CONTROL

The STB3 delay line, designed for JPL's Interferometry Technology Program, is a precise optical component with a very large dynamic range of motion. It achieves this by using three different position actuators with different properties. Figure 1 shows how the delay line works: Starlight enters the delay line cat's-eye and hits the parabolic mirror at the back, then bounces to the flat mirror on the PZT stack. Commands to the PZT make very small changes in the overall optical pathlength, but can be made at very high rates. The voice coil responds more slowly than the PZT but can move the cat's-eye as much as a centimeter. A stepper motor moves the entire cart along the track; it is the slowest actuator but has the largest range of motion. Together, the three actuators can respond to pathlength change commands with a dynamic range of 10^6.

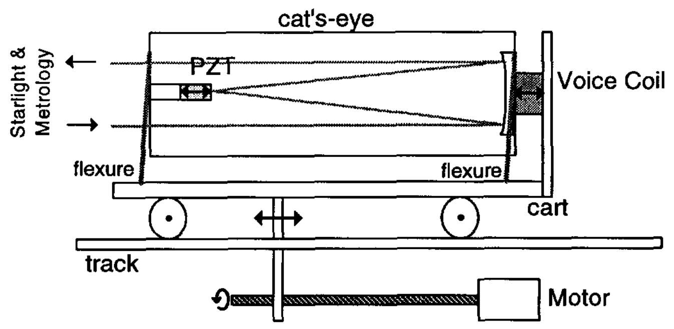

Figure 1: Delay Line Optical Component

The Delay Line control laws are organized into three partitions with six servo modes and two controller modes; it is the most complex servo in the STB3 system. However this does not mean that a Delay Line servo developer must write code for 36 different controllers! Only the motor partition has separate controllers for the four servo modes Front, Back, Creepfront, and Creepback, for example; the PZT and voice coil partitions simply re-use the Idle controllers for those two actuators. In addition, the controller modes only come into play when the servo mode is set to Track.

Each of the servo partitions runs at a rate that reflects the response time of its corresponding actuators:

PZT: (5000 Hz) Reads the current internal metrology value, compares it to the current target position, and commands the PZT to correct the error to the extent of its ability.

Voice Coil: (2500 Hz) Reads the PZT compensator input and calculates how the voice coil should move in order to prevent the PZT from saturating.

Motor: (100 Hz) Reads the Voice Coil command and calculates how the motor should move in order to prevent the Voice Coil from saturating. It also gets a feedforward velocity directly from the delay line target

Together, the three actuators cause a change in the optical pathlength (OPD), which alters the internal metrology measurement read by the PZT loop on the next servo cycle. Note the interaction between the control loops that is required to make the three position actuators work together smoothly (see also Figure 2).

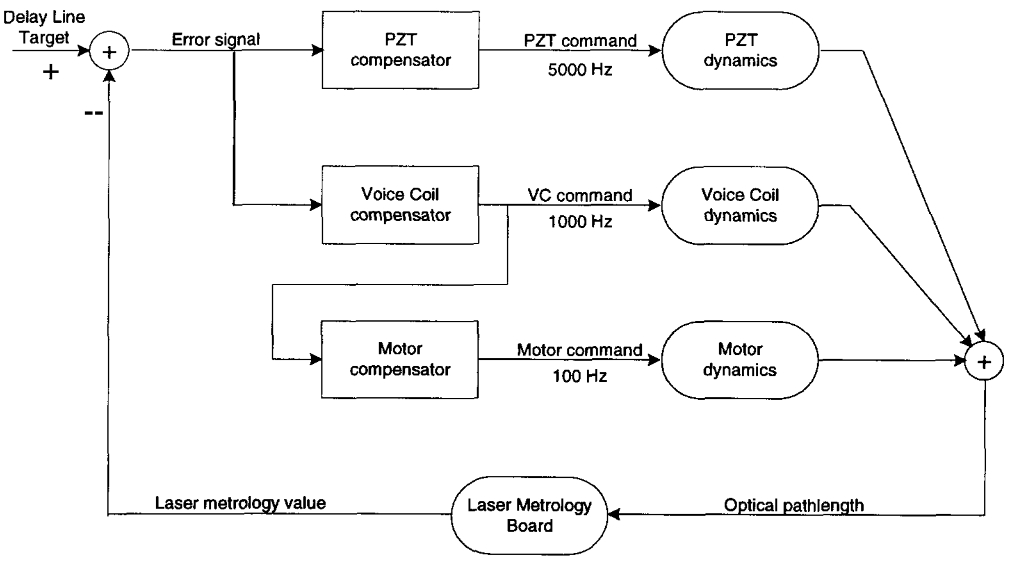

Figure 2: Delay Line Loop Interactions in Track Mode

Each partition has controllers for each of the following six servo modes.

Off: Reads internal metrology.
Front: Reads internal metrology and drives the motor at full speed forward (towards the front limit switch).
Back: Reads internal metrology and drives the motor at full speed backward (towards the back limit switch).
Creepfront: Reads internal metrology and drives the motor at slow speed forward.
Creepback: Reads internal metrology and drives the motor at slow speed backward.
Track: Reads internal metrology, calculates target position from sum of incoming target triples, drives the partition's actuator so that the difference between metrology value and target position goes to 0. Note that the way this is accomplished may change depending on the controller mode, see below.

Each partition, when in Track mode, will be in one of these two controller modes; however in practice the PZT and Voice Coil partitions are set up to simply reuse their "Off" controllers when in the Slew controller mode. Only the motor partition has a special Track-Slew controller implemented in the code.

Slew: Motor moves in direction of target position as fast as is reasonable (based on distance to be covered)
Track: Motor is only used as needed to offload voice coil.

When the servo mode is set to Track, the UpdateMode task running in the background monitors the position error, and changes controller mode automatically when it reaches a pre-determined level. When the error is sufficiently large, UpdateMode switches the controller mode to Slew, which idles the Pzt and Vc servos and causes the motor to move towards the target position. When the error drops below the given threshold value (there are actually two thresholds, to allow for hysteresis) UpdateMode switches the partitions to Track controller mode, where all three servos behave as described above.

From the perspective of the rest of the system, the delay line component behaves simply like a position actuator: Given a target value, it adjusts the optical pathlength accordingly and maintains it until commanded otherwise.

6. PHASING SUBSYSTEM: FRINGE TRACKING

Phasing consists of finding the fringe and then locking onto it. Finding the fringe is done open-loop, i.e. the optical path is varied according to a predetermined pattern, and the fringe SNR and visibility are measured until they reach preset threshold values. At this point the software automatically switches to a controller with a closed-loop control law to track the fringe.

The Fringe Tracker control laws are organized into three partitions with two servo modes and three controller modes. When the servo is in Track mode, the three partitions have the following behaviors:

APD: (5000 Hz) Outputs the APD photon counts read to a D/A channel as a diagnostic signal. This is normally attached to the input of a speaker. When the system is locked on a fringe, the variation in the counts produces a fairly clear sinusoidal tone from the speaker, with the quality of the tone reflecting the SNR of the fringe.

DelayLine: (1000 Hz) Calculates the fringe SNR, visibility, and phase; calculates targets to send to the Delay Line component to adjust the optical pathlength so that the fringe phase goes to 0.

Wobble: (75 Hz) Calculates "wobble" targets, used in comparing the quality of current fringe with adjacent ones (not currently done by STB3, but retained for future use).

Each partition has controllers for each of the following two servo modes.

Off: Counts photons & calculates fringe phase

Track: Counts photons & calculates fringe phase, calculates and sends targets to DL to change pathlength, either in search pattern or to send phase to 0

Each partition, when in Track mode, will be in one of these two controller modes; however in practice the APD and Wobble partitions have only one Track controller each, which they use for all three controller modes. Only the DelayLine partition has separate Track controllers for Search, Semi-Lock, and Lock.

Search: Targets sent to DL are calculated from user-defined search pattern algorithm

Semi-lock: Targets sent to DL are calculated to send phase to 0

Lock: Targets sent to DL are calculated to send phase to 0

When the servo mode is set to Track, the UpdateMode task running in background monitors fringe SNR & Visibility levels, and changes the controller mode when they reach certain thresholds. Semi-lock is only used briefly: It stops following the fringe search pattern but continues to monitor the fringe SNR & visibility, making sure the data represents an actual fringe and not just a glitch in the data. If the SNR and visibility remain sufficiently high, the controller mode is switched to Lock; otherwise it goes back to Search and continues the fringe search pattern. The logic performed by this task is shown in Figure 3.

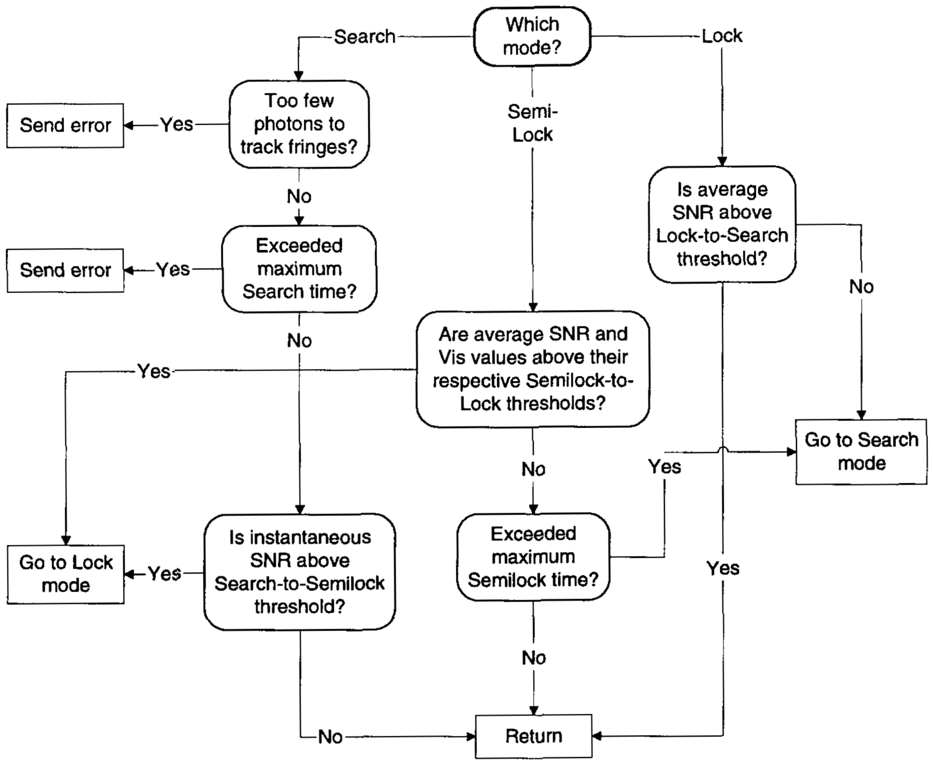
Which mode?
Search
Lock
Semi-Lock
Too few photons to track fringes?
Send error
Yes
No
Exceeded maximum Search time?
Send error
Yes
No
Is average SNR above Lock-to-Search threshold?
No
Yes
Go to Search mode
Are average SNR and Vis values above their respective Semilock-to-Lock thresholds?
Yes
No
Go to Lock mode
Is instantaneous SNR above Search-to-Semilock threshold?
Yes
No
Exceeded maximum Semilock time?
Yes
No
Return

Figure 3: Fringe Tracker: UpdateMode Task Logic

7. PATHLENGTH FEEDFORWARD

For 3-interferometer operation, the software on the guide interferometers is configured to send internal metrology and fringe phase information to the science interferometer, and the software on the science interferometer is configured to receive the data from the guides interferometers and to calculate how it should modify its own fringe tracking behavior to match what the guides are doing.

The Pathlength Feedforward (PFF) software component simply adds an open-loop offset to the delay line commands, with the assumption that disturbances corrected on the guide interferometers need to be corrected in the same way on the science interferometer. This offset is calculated from the pathlength measurements of the two guide interferometers, as well as their fringe phase measurements. Each interferometer's data is multiplied by a coefficient that represents its geometry. These coefficients need to be extremely accurate if high levels of disturbance rejection are to be achieved, as is illustrated later in this paper.

The control law implemented by the PFF servo is actually open-loop, since its outputs (the targets sent to the Science Delay Line component) have no effect on its inputs (the fringe phase and internal metrology coming from the Guide interferometers). It has only one partition, running at 1000 Hz, and in fact could be completely represented by a single controller (i.e. only one servo mode and one controller mode). For performance reasons, however, it uses two servo modes with slightly different behaviors:

Idle: Collects internal metrology and phase info from the Guide interferometers, and calculates the PFF target to send to the Science Delay Line component.

Track: Collects internal metrology and phase info from guides, the Guide interferometers, calculates the PFF target, and then sends it to the Science Delay Line.

This technique assumes that these higher-frequency disturbances affect all three interferometers equally, which is not strictly the case in reality. However great care has been (and is being) taken in the construction of the STB3 testbed so that (a) the three interferometers do see the same disturbances as much as possible, (b) where possible, the differences between the interferometers is measured and incorporated into the PFF algorithm [for example, external metrology beams will measure changes in the shape of the structure], and (c) any areas where the interferometers' responses might differ is recorded and analyzed for its potential effect on the data. Preliminary results on the optical tables indicate that the PFF technique works well in a controlled laboratory environment, with the guide interferometers tracking fringes from green laser "stars" and the science interferometer tracking fringes on a white light "star".

8. MEASURING PEFORMANCE: ATTITUDE CONTROL SYSTEM

SIM's requirement for STB3 is to be able to track the science fringe with 10nm maximum optical pathlength jitter at all frequencies. This includes a 6nm requirement at frequencies below 2Hz and 6nm at frequencies above 2Hz. The requirement above 2Hz has already been demonstrated by the Micro Precision Interferometer testbed[4], so STB3 has focused most of its efforts on achieving the desired fringe stability at frequencies below 2Hz. This is known as the "ACS regime" because the spacecraft's attitude control system is expected to generate disturbances in this frequency range.

Meeting this requirement has developed into a bigger challenge than was at first anticipated, because the laboratory environment of STB3, unlike SIM, contains atmosphere which contributes to the system noise under ambient conditions. It has been shown[6] that the ambient noise floor in the STB3 laboratory is no less than 45nm, making it impossible to meet SIM's jitter requirement by direct means. So instead of trying to demonstrate rejection of ambient disturbances down to the 10nm level, STB3 is working to show an equivalent level of rejection for signals of higher magnitude (i.e. significantly above the atmospheric noise floor). For this purpose, the anticipated on-orbit disturbance due to the spacecraft's attitude control system (ACS) is injected into the system.

For the PFF experiments, applying predefined motion to the pseudostar table simulates external disturbances. It is referred to as "Attitude Control" motion because it can be used to simulate disturbances originating from the spacecraft's attitude control system. The electronic and optical hardware design of the Attitude Control System (ACS) are described in more detail elsewhere[5], but basically the motion is generated by eight magnetic coils located at strategic points around and underneath the optical table. The system is capable of injecting motion in six degrees of freedom (i.e. X,Y,Z translation as well as rotation), at frequencies up to 10 Hz and as low as DC or constant offset, with magnitudes up to several millimeters. Results from experiments using disturbances provided by the ACS are shown in Section 10 below.

The Attitude Control component has a single 250 Hz partition with three servo modes:

Off: Measures the current table position.

Track: Measures the table position, calculates the target position, then commands coils to move the table to the target position. (In Fall of 2002 this will include closed-loop control, where the commands to the coils will incorporate compensation based on the error between the current table position and the target position.)

Home: Measures the table position and commands the coils with predefined values to hold table to constant position (open-loop).

The target position at a given time is calculated from an equation that represents a user-defined waveform. The Attitude Control component can generate a wide variety of target waveform types, including sinusoidal, square or triangle wave (with selectable duty cycle), Gaussian white noise, sine sweep ("chirp"), or filtered white noise (where the user can specify the filter coefficients).

9. DIM STAR FRINGE TRACKING

When observing a dim star, the photons received are drastically reduced and the control loops must be slowed down significantly in order to gather enough data to achieve good SNR and Visibility levels. At such a low rate, the servo control can compensate for only extremely low-frequency disturbances, such as a slow drift in the spacecraft attitude. Higher frequency vibrations generated by mechanical or electrical components of the spacecraft or the interferometer cannot be removed, and the fringe tracking will be very poor. So instead, the Science interferometer software counts on the information it receives from the two Guide interferometers (which are tracking fringes on bright stars at a much higher servo rate) in order to reject the higher-frequency disturbances as well. This is the PFF technique that has already been described. PFF was initially tested with the Science interferometer running completely open-loop, i.e. without using the fringe tracking control law at all. This successfully showed PFF rejecting the higher-frequency disturbances, but also revealed a small but persistent drift in the fringe position, too low in frequency to be corrected by the PFF algorithm.

In order to truly measure the performance of PFF, however, the fringe tracking behavior of the Science interferometer needed to be consistent with studying a dim star. The normal behavior of the Fringe Tracker component is to assume a sufficient quantity of photons are being received in order to close the fringe tracking servo loop at 1 kHz. This is appropriate for the Guide interferometers which are pointed at bright stars, but not for the Science. One solution might be to create an entirely new Fringe Tracker component encompassing the dim-star fringe tracking behavior, to be used exclusively on the Science interferometer. It turns out, however, that most of the behavior desired in the "dim star" Fringe Tracker is the same as what already exists in the "bright star" code. It is also more in keeping with the RTC Toolkit's emphasis on reusability to create instead a single Fringe Tracker component that can handle both types of fringe tracking. So the existing STB3 code was modified to change the fringe tracking behavior on command (with the default behavior being user-selectable in the system configuration).

The behavior in "dim star" mode is to effectively reduce the frequency at which the Fringe Tracker component updates its phase estimate, emulating SIM using a CCD camera for low-rate, low-SNR phase feedback during PFF. The fringe phase is still calculated from the photon counts at the usual 1 kHz rate but it is accumulated and averaged, and the updated average is only passed on to the Delay Line target calculation at 1 Hz (this rate is user-configurable). The averaged values are smoothed by filters to avoid injecting high frequencies into the target value. In addition, the SNR and visibility calculations are changed to reflect the longer "integration" time, and the cached values updated less often. If the controller mode is set to "Search", the search pattern targets are passed to the Delay Line at a lower rate as well.

It is worth noting that this feature could have been implemented by creating a new FT servo mode and putting the code for the new behavior into a new controller class. (Or perhaps a new controller mode would have been appropriate, with an UpdateMode task that monitored the number of photons, deciding whether the star was bright or dim and changing the mode accordingly.) In practice it turned out that the behavior changes could be implemented simply by changing filters, with all other behavior remaining the same, so it was easiest to implement the dim star "mode" as an additional variable in the data block shared by all the controllers. When the dim star mode is "on", the controllers use one set of filters; when it is "off" they do the same calculations using a different set. This is an example of the kind of design choice that a developer is free to make in using RTC; in this case the Fringe Tracker controller code was already in existence and in heavy use, so encapsulating the changes in a small region of the code greatly reduced the risk to the project schedule. If one were creating the FT gizmo from scratch, with full advance knowledge of all the required features, one might choose instead to implement the bright/dim star behavior as a separate servo or controller mode.

10. PERFORMANCE SUMMARY

A steady state survey of PFF performance (injecting single-frequency motion in each of the six degrees of freedom) has been conducted from 0.0005 Hz to 5.12 Hz. In addition, a random on-orbit like ACS disturbance has been synthesized and implemented in the testbed (99 % of power in frequencies < 1Hz). PFF rejection of this random on-orbit like ACS disturbance has been measured, which will be shown here as applied to Yaw (i.e. Z rotation). System performance in "dim star" fringe tracking mode shows rejection of drift and very low frequency dynamics (< 0.04).

In bright star fringe tracking mode the guide interferometers exhibit 40 dB/decade of external delay rejection from 30 Hz to DC. In path feed forward fringe tracking mode the Science interferometer can achieve the level of performance of the guide interferometers, but because of the atmosphere its noise floor is much higher.

Using PFF alone, i.e. tracking the fringe open-loop using data from the guides, the Science interferometer can achieve 75 dB rejection at 0.08 Hz to 86 dB at 0.005 Hz. Optimizing the PFF coefficients made a big difference (earlier rejection was only 65 dB); it turns out that it is necessary to know the PFF coefficients to at least the 5th decimal place in order to see this level of performance from the system.

Dim Star fringe tracking alone, where the fringe tracking loop is closed at a reduced rate but PFF is not used, exhibits performance levels of the guide interferometers only its bandwidth is 0.1 Hz. When dim star fringe tracking is applied in conjunction with PFF, however, the noise floor of the Science interferometer is largely reduced and its performance is improved below 0.1 Hz, eliminating the inevitable drift typical in open loop controllers such as PFF. Figure 4 summarizes the rejection achieved on single-frequency tests in yaw (i.e. Z-rotation) at various frequencies.

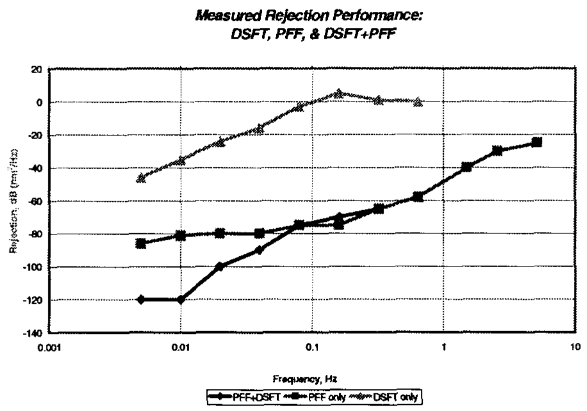

Figure 4: Rejection Performance of PFF and DS Fringe Tracking

11. ACKNOWLEDGEMENTS

The research described in this publication was performed at the Jet Propulsion Laboratory of the California Institute of Technology, under contract with the National Aeronautics and Space Administration.

12. REFERENCES

1. See the SIM website at http://sim.jpl.nasa.gov/whatis/
2. R. L. Johnson, Jr., E. A. McKenney, and K. M. Starr, "Real-time Control Software for Optical Interferometers: The RICST Testbed", *SPIE Conference on Interferometry in Optical Astronomy*, **3350**, pp. 153-162, 1998.
3. T. Lockhart, "RTC – A Distributed Realtime Control System Toolkit", *These Proceedings*, paper 4848-21, 2002.
4. R. Goullioud, F. G. Dekens and G. W. Neat, "Micro- Precision Interferometer: Scoreboard on technology Readiness for the Space Interferometer Mission", *SPIE Conference on Interferometry in Optical Astronomy*, **4006**, pp. 847-858, 2000.
5. Y. Gürsel and E. McKenney, "Attitude Control System for the SIM interferometry testbed 3 (STB3)", *These Proceedings*, paper 4852-69, 2002.
6. O. S. Alvarez-Salazar, R. Goullioud, B. Nemati, and A. Azizi, "Path feed forward performance of an astrometric 3-BL interferometer testbed (STB-3)", *These Proceedings*, paper 4852-89, 2002.